Reducing the Cost of Large Register Files in EPIC Architectures with Stacked Register Aliasing

Ron Arnold, Rohit Bhatia, Don Soltis

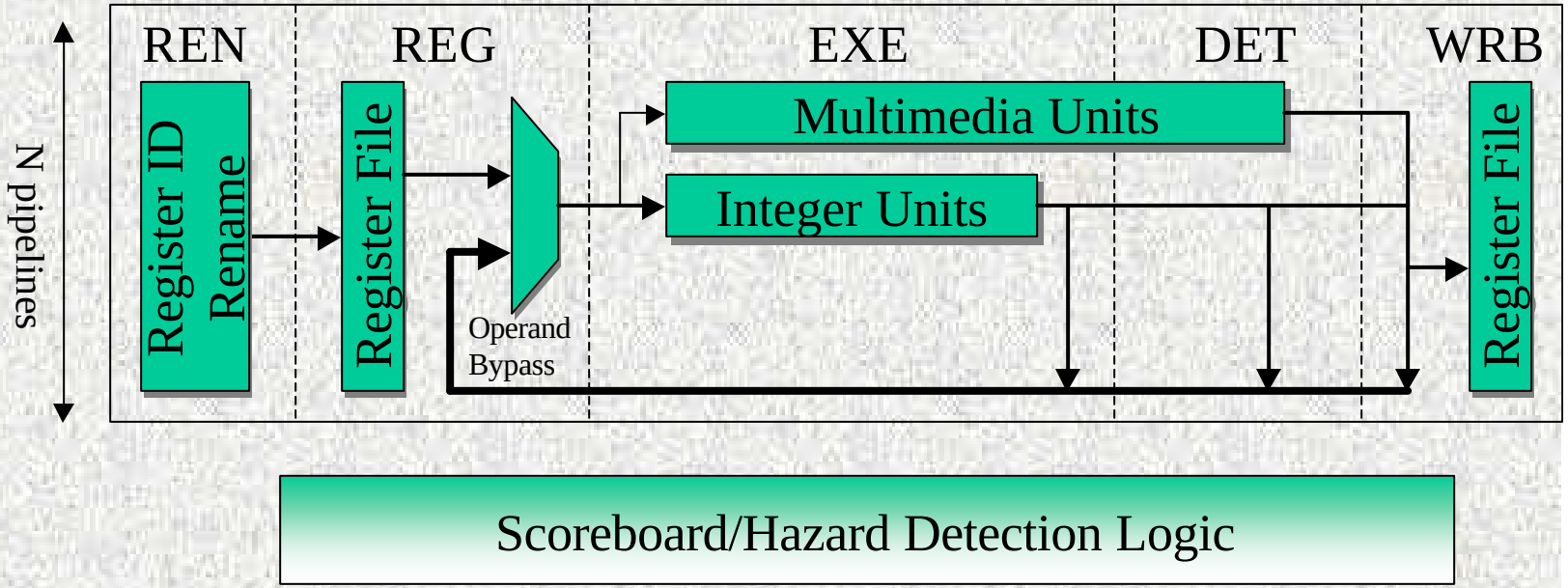
EPIC-2, November 2002

Stacked Register Aliasing

- Large register files are a key ingredient in EPIC performance delivery
  - ILP enhancement achieved thru speculation, predication
  - Software pipelining with mod-sched loops
  - Register stacking for procedure call/return
- Itanium® arch has only 0.71x mem ops (incl RSE) compared to Alpha[1]
  - RSE cycles are only 3-4% of total cycles

1. McCormick et al. HotChips 2002

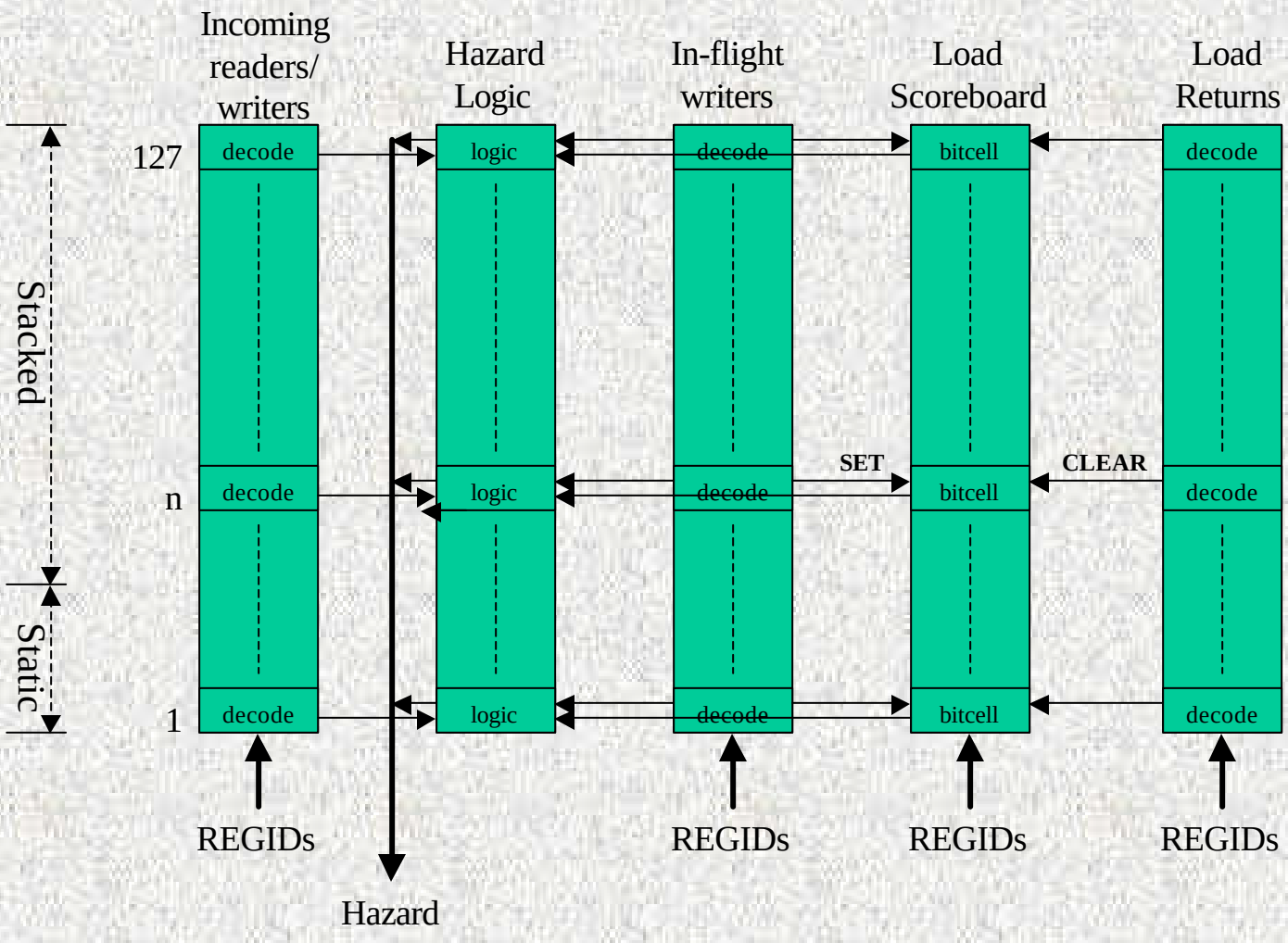
Stacked Register Aliasing



Complexity Factors
- Issue width or number of pipelines
- Number of pipeline stages
- Number of physical registers
- Instruction qualifiers

Stacked Register Aliasing

Hazard Logic Array Structure



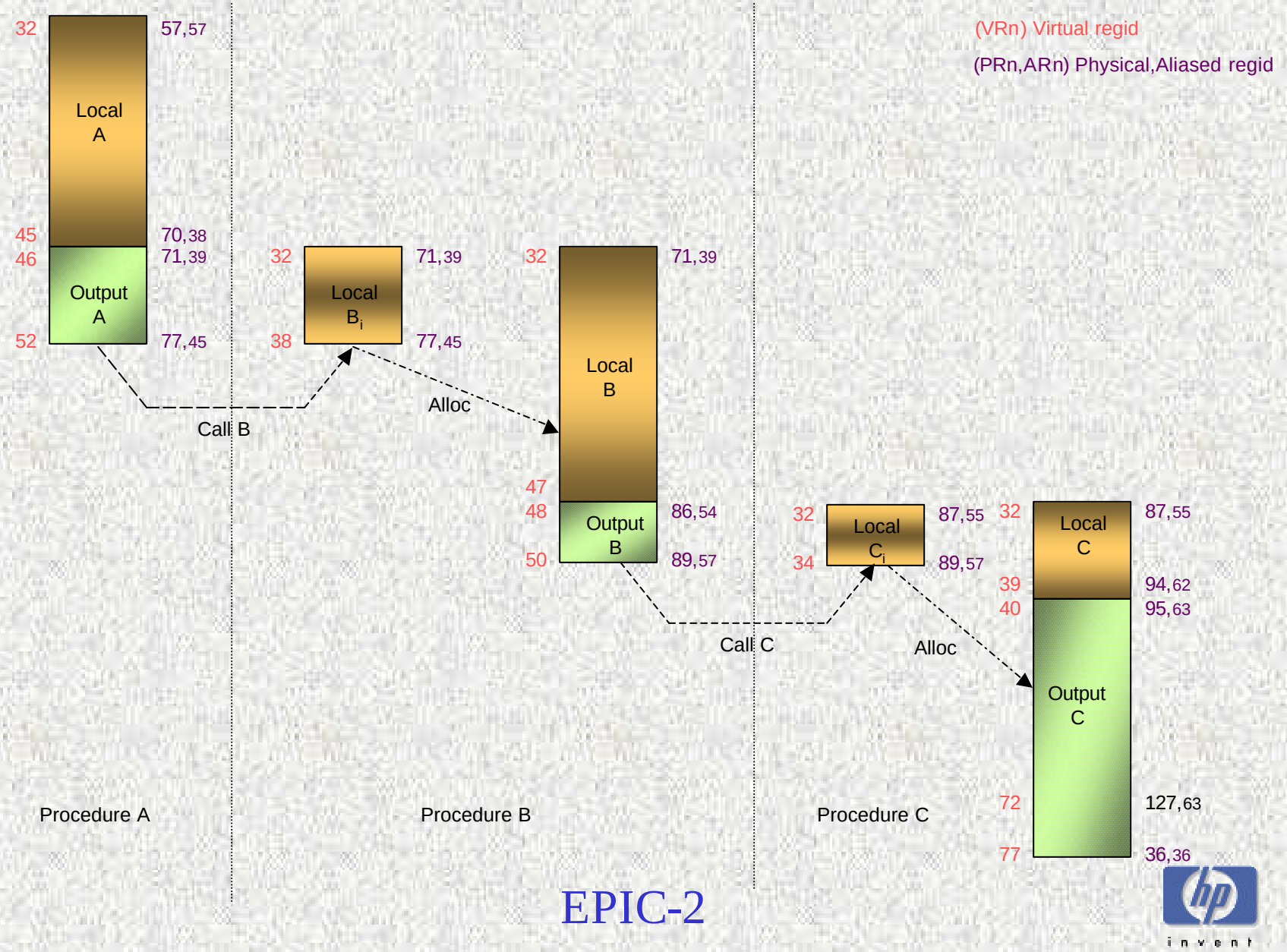
EPIC-2

Stacked Register Aliasing

- Hazard Logic Array(HLA) structure already simplifies non-linear dependence on number of pipelines and pipeline stages
  - However, number of rows in the HLA is still a 1-1 mapping to number of registers
- Static registers are always accessible, but stacked registers must be allocated prior to use
  - Only a fraction of the stacked registers are actually used at any given time
- ***Register Aliasing*** maps multiple physical registers onto a single row of the HLA
  - ***Stacked register aliasing*** partitions the HLA into static register rows and stacked register rows
  - For example, a HLA with 64 total rows might be partitioned into 32 rows for the 32 static registers and 32 rows for the remaining 96 stacked registers
  - Now, GR[32], GR[64] and GR[96] would all map to the same row in the HLA i.e. 3-way aliasing

Stacked Register Aliasing

Example Call Sequence



EPIC-2

invent

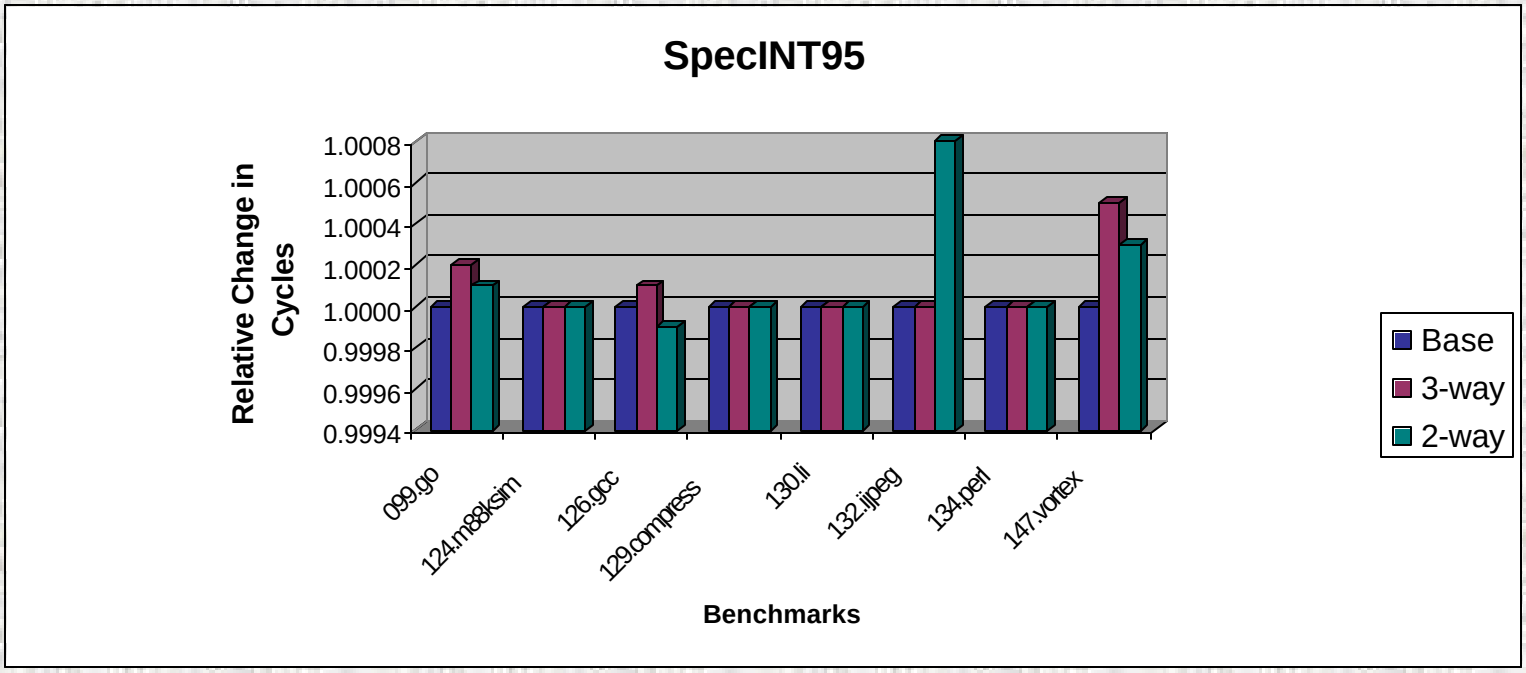
Stacked Register Aliasing

- Reduces the physical design complexity of hazard detection
  - Allows a power-performance tradeoff
- Decouples register file design decisions from hazard detection logic
  - Physical registers can be increased without additional complexity in hazard detection
- False hazard stalls can occur resulting in a performance penalty

Stacked Register Aliasing

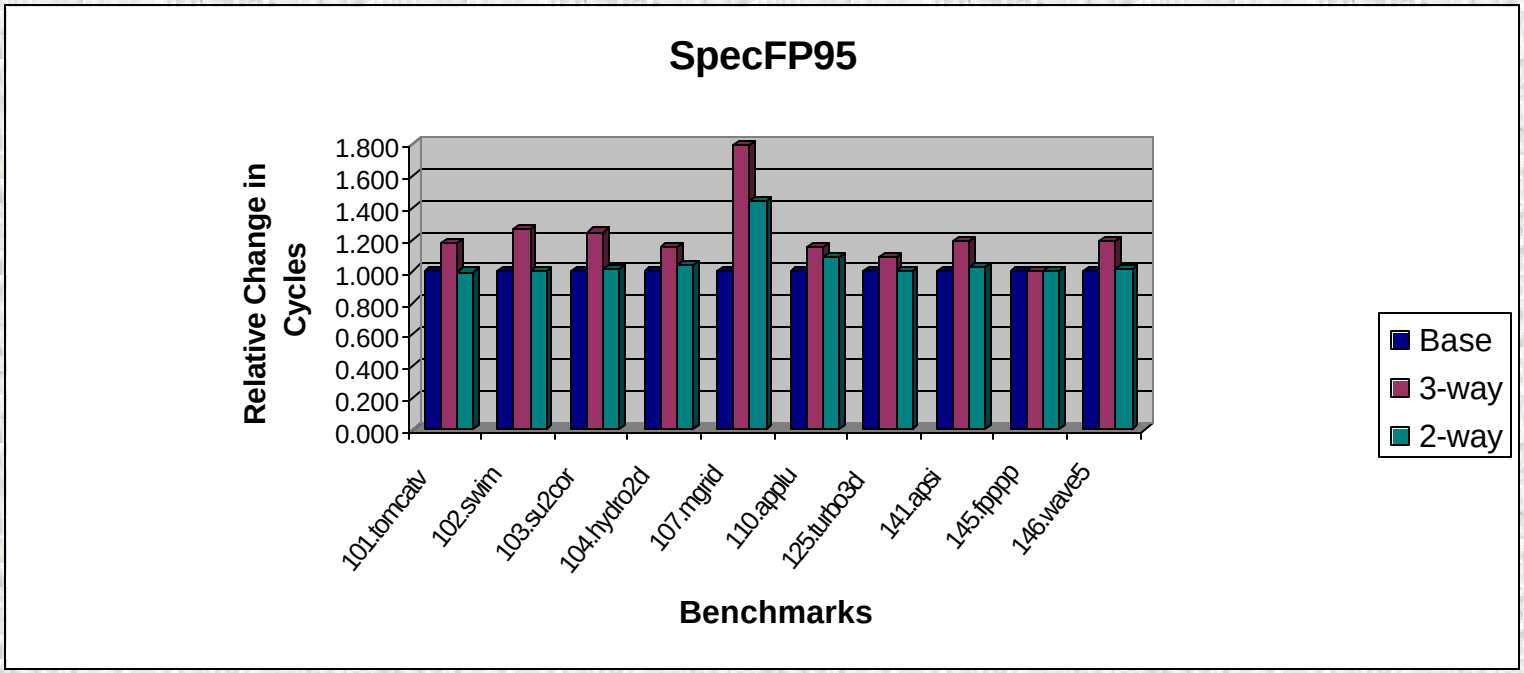
- All benchmarks compiled with internal HP IPF compilers
- An internal performance simulator which models the Itanium® 2 microarchitecture was used for the study
- Three sets of runs were performed to capture total cycle counts
  - Without aliasing, to capture a baseline
  - 2-way and 3-way aliasing, after tweaking the simulator to implement aliasing in hazards

Stacked Register Aliasing



Negligible performance loss
- Little use of mod-sched loops
- Compiler scheduling for single cycle 1st level cache

Stacked Register Aliasing



Some performance loss, decreases from 3-way to 2-way aliasing

- Heavy use of mod-sched loops
- Compiler scheduling for 2nd level cache

Stacked Register Aliasing

- Results suggest that stacked register aliasing can be implemented for integer registers
  - When combined with increased physical registers, can probably achieve an overall performance gain
- For floating-point registers, register aliasing provides a technique to make power-performance tradeoff
  - Also, in Itanium® arch, FP registers only perform rotation and not stacking

Stacked Register Aliasing

- Update with Spec2K benchmarks
  - Use other workloads e.g. TPCC, SpecJBB
- Integrate register aliasing with RSE benefits for additional physical registers
  - Have initial data suggesting performance gains achievable with additional 32 physical registers due to reduction in RSE traffic
  - Combine the two schemes and study overall performance improvements

Acknowledgements

Thank You

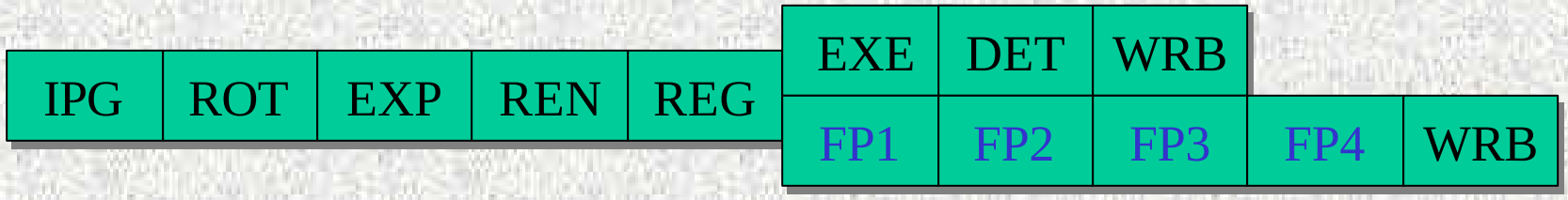
Greg Woods, Subramoni Parmeswaran and Terry Lyon for helping in setup and collection of performance data.

MICRO-35/EPIC-2 Program Committee and John Crawford for reviewing and providing valuable feedback.


hp
invent

Backup Slides

Stacked Register Aliasing



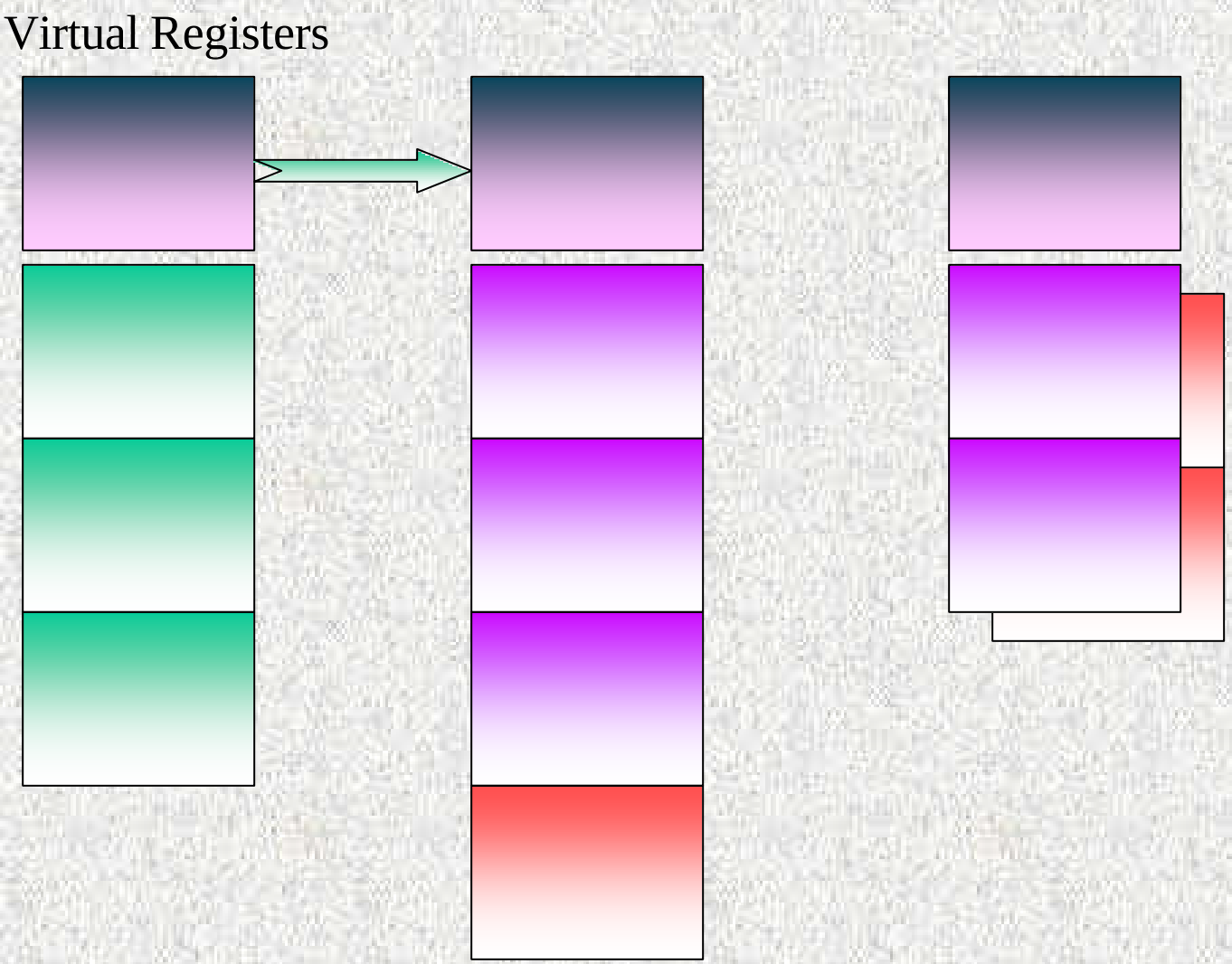
IPG: Instruction Pointer Generate, Instruction address to L1 I-cache
ROT: Present 2 Instruction Bundles from L1 I-cache to dispersal hardware
EXP: Disperse up to 6 instruction syllables from the 2 instruction bundles
REN: Rename (or convert) virtual register IDs to physical register IDs
REG: Register file read, or bypass results in flight as operands

EXE: Execute integer instructions; generate results and predicates
DET: Detect exceptions, traps, etc.

FP1-4: Execute floating point instructions; generate results and predicates

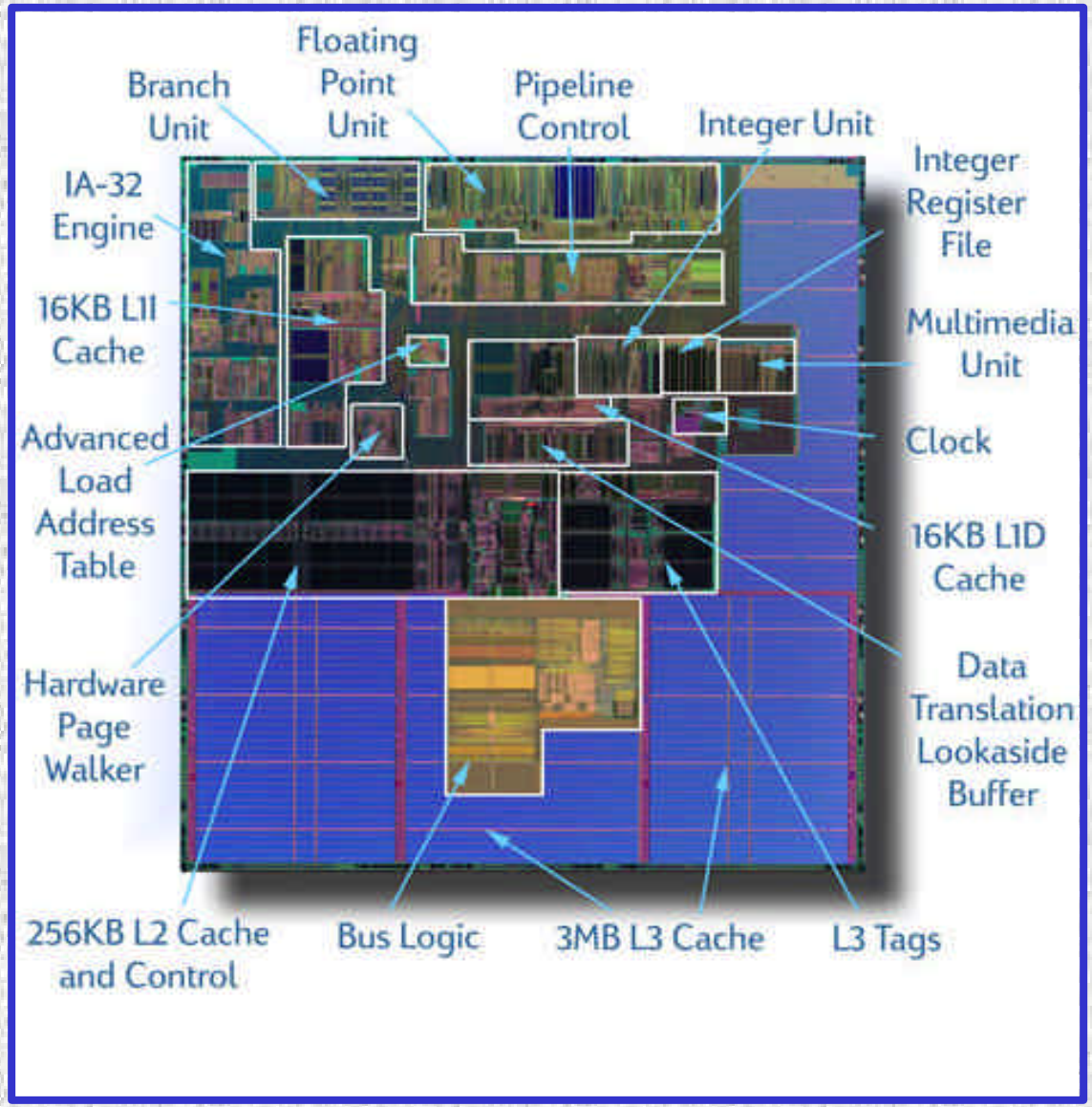
WRB: Write back results to the register file (architectural state update)

Stacked Register Aliasing



Stacked Register Aliasing

Itanium® 2 Die Photo



EPIC-2

invent